CLAIMS

What is claimed is:

1. A system that facilitates processing rules, comprising a translator component that translates synchronous statements to asynchronous instructions using a synchronous programming model.

2. The system of claim 1, the statements are organized into a series of rule types.

3. The system of claim 2, the rule types express logic that determines the desired state of a target resource.

4. The system of claim 3, in response to a desired state, and action is taken.

5. The system of claim 1, the translator component facilitates instantiation of the asynchronous instructions.

6. The system of claim 1, the instructions facilitate concurrent processing by a runtime engine.

7. The system of claim 6, the instructions facilitate maintaining all states by the runtime engine, which states include at least one of arguments and local variables.

8. The system of claim 1, the instructions facilitate at least one of yielding to runtime rule code execution switching and calling a utility function.

9. The system of claim 1, the instructions insert periodic yield statements.

10. The system of claim 1, the translator component facilitates depth-first traversal to generate at least one of labels and temporary variables for corresponding nodes.

11. The system of claim 1, the translator component translates modules of the rules into classes.

12. The system of claim 1, the instruction includes an address code representation to facilitate context switching by a runtime engine.

13. The system of claim 12, the address code representation employs at least three address codes, which representation is used when a statement or expression includes an asynchronous call.

14. The system of claim 1, the instructions are translated into a series of instruction blocks that are separated into switch-case blocks.

15. The system of claim 14, instructions within the instruction block are executed as a unit, and yield to runtime execution only between such instruction blocks.

16. The system of claim 14, the series of instruction blocks has associated therewith an update identifier that maintains which instruction block of the series is to be executed next upon reentry into the rule.

17. A system that facilitates concurrent processing of rules, comprising:

a translator component that translates the rules into instructions for concurrent processing; and

a runtime engine that schedules the instructions for processing and processes some or all of the instructions concurrently according to the schedule.

18. The system of claim 17, the runtime engine facilitates implicit concurrent processing of the rules.

19. The system of claim 17, the runtime engine receives both the instructions and configuration data, which configuration data specifies at least one of which rules to run and the parameters required to run the rule.

20. A system that facilitates processing rules in a model-based management architecture, comprising:

a plurality of rules that express health criteria for the architecture;

a translator component that translates the rules into asynchronous instructions for concurrent processing; and

a runtime engine that schedules the translated instruction for processing and processes some or all of the instructions concurrently according to the schedule.

21. The system of claim 20, the runtime engine operates according to one of the instructions that suspends processing of code and waits for an event to occur, in response to which processing of the code is resumed and allowed to act on the event.

22. The system of claim 20, one of the plurality of rules invokes another rule.

23. The system of claim 20, the engine schedules execution of an execution stack based on a polling structure for the current frame and if the polling structure is at the top of the execution stack.

24. The system of claim 20, the instruction component includes a language that facilitates looping, such that prior to jumping, a rule returns to the runtime engine to facilitate non-pre-emptive scheduling.

25. The system of claim 20, the translated instructions facilitate cooperative multitasking so that an execution state of a function is preserved, and constituted at a later time.

26. The system of claim 20, the instruction facilitate constructing a call frame, which call frame includes at least one of a function parameter, current instruction block, current function, and local variable.

27. The system of claim 20, the runtime engine includes a spreading algorithm that spreads execution of tasks over a duration of time beginning from an arbitrary time to reduce over-utilization.

28. A computer system according to the system of claim 20.

29. A method of processing rules, comprising:

receiving a plurality of the rules;

translating the rules into instructions for communication to a runtime engine;

scheduling the translated instructions for processing by the runtime engine;

processing at least two of the plurality of instructions concurrently with the runtime engine; and

receiving configuration data into the runtime engine to instantiate the rules.

30. The method of claim 29, further comprising injecting yield instructions into the rule to facilitate yielding execution of the rule to rule code execution switching during processing by the runtime engine, and to facilitate calling utility functions provided by the runtime engine.

31. The method of claim 29, the translated instructions are translated into a series of instruction blocks, wherein commencement of execution of the instruction block causes the instruction block to be executed in its entirety before yielding to the runtime engine.

32. The method of claim 29, further comprising during a startup process, building a task list of rules marked for execution by the runtime engine at startup, and spreading out initiation of the rules in batches such that each batch of rules is scheduled for processing within a time configured time interval.

33. The method of claim 32, further comprising configuring the time interval based upon predetermined criteria, which criteria includes a number of processors utilized by the computer.

34. The method of claim 29, the instructions are scheduled for processing in batches by the runtime engine, such that rules of a batch are scheduled for processing uniformly over associated time interval for processing the batch.

35. The method of claim 29, further comprising scheduling the translated instructions for processing based upon a polling structure, which processing occurs for at least one of the polling structure of a current frame and the polling structure at the tops of a stack.

36. The method of claim 29, scheduling is performed according to a non-preemptive scheduling regime.

37. A system for processing rules, comprising:

means for receiving a plurality of the rules;

means for converting the rules into instructions;

means for translating the instructions for processing by a runtime engine; and

means for scheduling and processing concurrently at least two of the plurality of instructions.

38. The system of claim 37, the means for converting includes means for inserting control relinquishment code into the rule that determines where to relinquish control of the associated translated instruction during execution thereof, which relinquishment code is associated with a stack frame that contains at least one of parameters, local variables, and where in the translated instruction to jump upon reentry.

39. The system of claim 37, further comprising means for task switching between rules, where the translated instructions includes call frames to facilitate task switching.

40. A computer-readable medium having computer-executable instructions for performing concurrent processing of rules, comprising a translator component that translates synchronous rule statements into asynchronous instructions using a synchronous programming model, which instructions facilitate calling utility functions and yielding to runtime code switching.